cdscan and *CDML* – file aggregation

The “cdscan” utility (1)

- CDAT provides a command-line file aggregation utility is called ***cdscan***.
- This allows you to describe an entire dataset with just one XML file, that is opened by CDAT using the standard `cdms.open()` call.
- The XML format is known as ***Climate Data Markup Language (CDML)*** which is fully described in the CDAT manual.
- Using CDML files:
  - removes the need to know about filename
  - provides a global description of a collection of files
  - metadata and aggregation are handled together

CDML structure

- CDML files contain the following sections:
  - <dataset> - general information at the dataset level.
  - <axis> - axis dimension information.
  - <variable> - relating to individual variables.

- ***At BADC we use and ECMWF ERA-40 CDML file which:***
  - ***links to over 3,000,000 files***
  - ***is only 21KB in size!***

cdscan in action

- **cdscan** will analyse the archive for:
  - variable information
  - axis information
  - global (universal) metadata

- Let’s have a look at it in action:
  - 1200 monthly mean NetCDF files to be scanned.
  - Scenario 1: Filenames do not map nicely to their contents. So we run cdscan plain and see what comes out.

```
$ cdscan –x monthly_means.xml ./*.nc
```

Using templates for filenames

- Scenario 2: Filenames reflect the contents of the files closely with the file-naming convention:

  ```
  <YYYY><MM>_<VARIABLE>.nc
  ```

- In the olden days, cdscan used to be "**cdimport**" which had one excellent feature you might want to make use of. It allows you to add a template for file and directory names.

- The template allows you to specify time components, start and end levels as well as variable IDs.

“cdimport”: cdscan’s predecessor

```
$ cdimport –h # yields information about the template:

%d day number (1 .. 31)
%eX ending timepoint/level, where X is a specifier character
%f day, two-digit, zero-filled (01, 02,…, 31)
%g month, lower case, three characters ('jan', 'feb', ...)
%G month, upper case, three characters ('JAN', 'FEB', ...)
%H hour (0 .. 23)
%h hour, two-digit, zero filled (00, 01, …, 23)
%L vertical level (integer)
%m month number, not zero filled (1 .. 12)
%M minute 0 .. 59
%n month number, two-digit, zero-filled (01, 02, ..., 12)
%S second (0 .. 59)
%v variable ID (string)
%y year, two-digit, zero-filled (integer)
%Y year (integer)
%z Zulu time (ex: '6Z19990201')
%% percent sign
```

Back to the example

- Scenario 2: Filenames reflect the contents of the files closely with the file-naming convention:

```
<YYYY><MM>_<VARIABLE>.nc
```

- Run cdscan with the –p argument and your template:

```
$ cdscan –x monthly_means.xml -p %Y%n_%v.nc /*.nc
```

- Optionally, you can do a manual edit of the XML file to tidy up the unused <cdms_filemap> attribute.

- This may hold millions of elements if you have a lot of files which makes it slow to read.

What else can cdscan do? (1)

- Let’s look at the help output from “cdscan –h”:

-a alias_file: change variable names to the aliases defined in an alias file.

-c calendar: either "gregorian", "proleptic_gregorian", "julian", "noleap", or "360_day". Default:

-d dataset_id: dataset identifier. Default: "none"

-e newattr: Add or modify attributes of a file, variable, or axis.

What else can cdscan do? (2)

--exclude var,var,...: exclude listed variables from output.

-f file_list: file containing a list of absolute data file names, one per line.

-h: print a help message.

-i time_delta: scan time as a 'linear' dimension. This is useful if the time dimension is very long.

--include var,var,...: only include the listed variables in the output.

What else can cdscan do? (3)

-j: scan time as a vector dimension. Time values are listed individually. Turns off the -i option.

-l levels: list of levels, comma-separated. Only specify if files are partitioned by levels.

-m levelid: name of the vertical level dimension. The default is the name of the vertical level dimension.

-p template: Compatibility with pre-V3.0 datasets. 'cdimport -h' describes template strings.

-q: quiet mode

What else can cdscan do? (4)

-r time_units: time units of the form "<units> since yyyy-mm-dd hh:mi:ss", where <units> is one of "year", "month", "day", "hour", "minute", "second“.

-s suffix_file: Append a suffix to variable names, depending on the directory the data is located in, deals with multiple files holding variables with the same name.

What else can cdscan do? (5)

-t timeid: id of the partitioned time dimension. The default is the name of the time dimension.

--time-linear tzero,delta,units[,calendar]: Override the time dimensions(s) with a linear time dimension. The arguments are a comma-separated list.

-x xmlfile: XML filename. By default, output is written to standard output.

So what does the user see?

- **cdscan**ned files are same as any other CDAT-compatible data file:

```
>>> import cdms
>>> f=cdms.open(‘cdscanned_stuff.xml’)
>>> print f.variables # Will list the
  variables
>>> var=f(‘q’, time=(“1910-10”, “1940-09”),
    lat=(30,60), lon=(-20,10), level=1000)
# var now holds the contents of whatever
# actual data files needed to be aggregated
# together.
```

- As a user you see none of this and can get on with your science!

So why use cdscan?

1. Large datasets described as a grouped entity.
2. No need to know underlying data format.
3. No need to know file-names.
4. Datasets can be sliced in any way the user chooses using logical spatio-temporal selectors rather than loops of programming code.
5. You can use it to improve the metadata of your data files…

cdscan to up your metadata quality!

- Since cdscan exposes a common set of metadata for a dataset it can be used to *improve your CF-compliance*!
- Use the ‘-e’ argument to add new attributes to your variables, axes and at the global file level:

```
-e temp.standard_name=“air_temperature”
-e temp.units=“K”
-e level.standard_name=“depth”
-e .source=“UK Met Office Unified Model Version 5.5”
-e .references=“Cited in paper by E.S.Fuller (2001).”
```